Generating Concise Entity Matching Rules

Rohit Singh[†] Vamsi Meduri[‡] Ahmed Elmagarmid[⋆] Samuel Madden[†]
Paolo Papotti[‡] Jorge-Arnulfo Quiané-Ruiz[⋆] Armando Solar-Lezama[†] Nan Tang[⋆]

[†]CSAIL, MIT, USA [‡]Arizona State University, USA
[⋆]Qatar Computing Research Institute, HBKU, Doha, Qatar

{rohitsingh, madden, asolar}@csail.mit.edu, {vmeduri, ppapotti}@asu.edu
{aelmagarmid, jquianeruiz, ntang}@hbku.edu.qa

ABSTRACT

Entity matching (EM) is a critical part of data integration and cleaning. In many applications, the users need to understand why two entities are considered a match, which reveals the need for *interpretable* and *concise* EM rules. We model EM rules in the form of *General Boolean Formulas* (**GBF**s) that allows arbitrary attribute matching combined by conjunctions ($\bigvee$), disjunctions ($\bigwedge$), and negations ($\neg$). GBFs can generate more concise rules than traditional EM rules represented in disjunctive normal forms (**DNF**s). We use *program synthesis*, a powerful tool to automatically generate rules (or programs) that provably satisfy a high-level specification, to automatically synthesize EM rules in **GBF** format, given *only* positive and negative matching examples.

In this demo, attendees will experience the following features: (1) *Interpretability* – they can see and measure the conciseness of EM rules defined using **GBF**s; (2) *Easy customization* – they can provide custom experiment parameters for various datasets, and, easily modify a rich predefined (default) synthesis grammar, using a Web interface; and (3) *High performance* – they will be able to compare the generated concise rules, in terms of accuracy, with probabilistic models (e.g., machine learning methods), and hand-written EM rules provided by experts. Moreover, this system will serve as a general platform for evaluating different methods that discover EM rules, which will be released as an open-source tool on GitHub.

1. INTRODUCTION

Entity matching (EM), where a system or user finds records that refer to the same real-world object, is a fundamental part of data integration and data cleaning.

There is a key tension in EM algorithms: On one hand, algorithms that properly match records, i.e., *high accuracy*, are clearly preferred. On the other hand, algorithms need to be *interpretable* – that is, the user of the system needs to understand *why* two entities are considered a match. Systems that use probabilistic models – such as machine learning methods based on SVMs [2], or fuzzy matching [4] – are much harder to interpret and hence are often not preferred in applications that handle critical data such as healthcare. In contrast, systems that are rule-based ("deterministic") [3] offer better interpretability, particularly when the rules can be constrained to be simple (i.e., consist of relatively few clauses). A key question, however, is whether such simple rules can match the effectiveness of probabilistic approaches while preserving interpretability.

Although hand-writing EM rules may be practical in some limited domains, doing so is extremely time consuming and error-prone. Hence, a promising direction is to use automatic methods to generate deterministic EM rules, e.g., by learning rules from training examples. This learning should be done with as few examples as possible, because generating the training examples is itself laborious too.

We have developed a system that can effectively learn EM rules that (i) match the performance of probabilistic methods, (ii) produce concise and interpretable rules, and (iii) learn rules from limited training examples. Our approach is to use *program synthesis* [5] (PS), in which a program (set of rules) is generated by using positive and negative examples as constraints that guide the synthesizer towards rules that match the examples. The demo will demonstrate the following three key features: (1) *Interpretability*, (2) *Easy customization*, and (3) *High performance*.

SIGMOD '17, May 14–19, 2017, Chicago, IL, USA.
© 2017 ACM. ISBN 978-1-4503-4197-4/17/05…$15.00
DOI: http://dx.doi.org/10.1145/3035918.3058739

2. RULES AND ALGORITHMS

Let $R[\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_n]$ and $S[\mathsf{A}'_1, \mathsf{A}'_2, \ldots, \mathsf{A}'_n]$ be two relations with corresponding sets of n aligned attributes A_i and A'_i ($i \in [1, n]$). As in previous work [6], we assume that the attributes between two relations have been aligned and provided as an input; this can either be manually specified by the users, or done automatically using *off-the-shelf* schema matching tools [1]. Let r, s be records in R, S and $r[\mathsf{A}_i], s[\mathsf{A}'_i]$ be the values of attributes $\mathsf{A}_i, \mathsf{A}'_i$ in records r, s, respectively.

A *similarity function* $f(r[\mathsf{A}_i], s[\mathsf{A}'_i])$ computes a similarity score in the real interval $[0, 1]$. A bigger score means that $r[\mathsf{A}_i]$ and $s[\mathsf{A}'_i]$ have a higher similarity. Examples of similarity functions are cosine similarity, edit distance, and Jaccard similarity. A library of similarity functions $\mathcal{F}$ is a set of such general purpose similarity functions, such as the Simmetrics (https://github.com/Simmetrics/simmetrics) Java package.

Attribute-Matching Rules. An *attribute-matching rule* is a Boolean predicate representing $f(r[\mathsf{A}_i], s[\mathsf{A}'_i]) \geqslant \theta$, where $i \in [1, n]$ is an index, f is a similarity function

	#-Postive Examples	#-Negative Examples	#-Attributes
D_C	14, 280	170, 379	9
D_{AG}	1, 300	95, 707	4
D_{LF}	6, 048	335, 196	10
D_C = **Cora**, D_{LF} = **Locu-FourSquare** D_{AG} = **Amazon-GoogleProducts**			

Table 1: Dataset Satistics

and $\theta \in [0, 1]$ is a threshold value. Attribute-matching rule $f(r[\mathsf{A_i}], s[\mathsf{A_i'}]) \geqslant \theta$ evaluating to *true* means that $r[\mathsf{A_i}]$ *matches* $s[\mathsf{A_i'}]$ relative to f and θ.

Notation. Let $\mathsf{f}[\mathsf{A_i}] \geqslant \theta$ be an attribute-matching rule $f(r[\mathsf{A_i}], s[\mathsf{A_i'}]) \geqslant \theta$ since $\mathsf{A_i}$ and $\mathsf{A_i'}$ are known to have been aligned. For example, $\mathsf{Soundex[title]} \geqslant 0.937$ is an attribute-matching rule that applies Soundex similarity function on the title attribute from R and the name attribute from S (omitted in the notation), and, compares it with the threshold 0.937. Note that for brevity we refer to attribute-matching rules as *atoms* of a larger Matching Rule.

Boolean Formula Matching Rule. A *Boolean formula matching rule* is an arbitrary *Boolean formula* with attribute-matching rules as its variables (or *atoms*) and conjunction ($\bigwedge$), disjunction ($\bigvee$) and negation ($\neg$) as allowed operations. Some Boolean Formula Matching Rules are:

$$\begin{aligned}
\varphi_1 &: (\mathsf{Levenstein[title]} \geqslant 0.8 \bigwedge \mathsf{Equal[volume]} \geqslant 1.0 \\
&\quad \bigwedge \mathsf{Equal[pages]} \geqslant 1.0) \\
\varphi_2 &: (\ \mathsf{Levenstein[title]} \geqslant 0.5 \bigwedge \mathsf{Equal[author]} \geqslant 1.0) \\
\varphi_3 &: (\ \varphi_1 \bigvee \varphi_2\ ) \\
\varphi_4 &: \textbf{if } (\mathsf{noNulls[author]} \geqslant 1.0) \textbf{ then } \varphi_2 \textbf{ else } \varphi_1
\end{aligned}$$

Note that "**if** (u) **then** (v) **else** (w)" can be expressed with the help of the negation operator $(\neg)$ as $(u \bigwedge v) \bigvee (\neg u \bigwedge w)$ but the "**if then else**" representation is more interpretable and concise. This is where we use the power of program synthesis to find a *General Boolean Formula* (**GBF**) that represents a *Boolean Formula matching rule*.

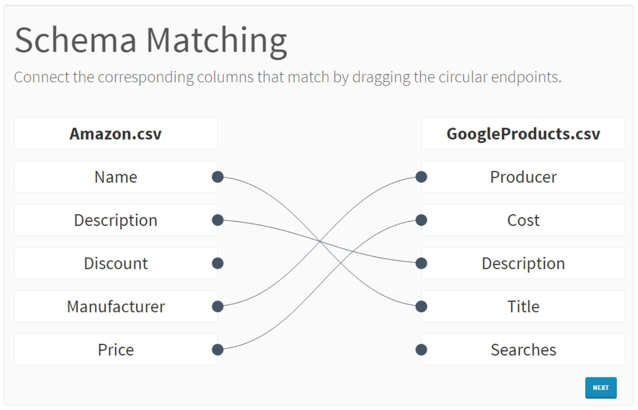
Synthesis Algorithm. Our novel synthesis algorithm searches for **GBF**s from a rich interpretable *grammar* (Figure 4) using a small set of matching and non-matching examples as constraints while smartly expanding the set to include important corner cases. This process is repeated multiple times to account for noise in the provided data and the best **GBF** is chosen across all generated **GBF**s by maximizing an *optimization metric* (e.g. F-measure).

3. DEMONSTRATION OVERVIEW

In this demonstration, we will show how easy it is for the users to obtain concise EM rules with our system without tuning parameters (Default Configuration). We will show the performance of our system compared with other *state-of-the-art* techniques (Evaluation). In addition, we allow users to customize the system to suit their needs (Customization). We will focus on the EM application and show the underlying PS problems off-line if there is interest.

Datasets. Table 1 shows three real-world datasets to be used in this demonstration. We prune the Cartesian product of records (comprising of up to 400 million pairs) from positive examples to construct negative examples.

Dataset Specification. The schemas of two relations in the datasets are aligned either by *off-the-shelf* tools, or by the user using our Web interface, as shown in Figure 1.


Schema Matching

Connect the corresponding columns that match by dragging the circular endpoints.

Figure 1: Schema Alignment

Moreover, the specification should also include a set of positive (and negative) matching examples as a file. Table 2 shows one positive matching example for the **Cora** dataset. The attendees can select one of the pre-specified datasets (Table 1) and generate a concise EM rule for that dataset:

$$\begin{aligned}
\varphi_{synth} :\ &(\ \mathsf{ChapmanMatchingSoundex[author]} \geqslant 0.937 \\
&\bigwedge\ \textbf{if} \quad \mathsf{noNulls[date]} \geqslant 1 \\
&\quad\ \ \textbf{then} \quad \mathsf{CosineGram_2[date]} \geqslant 0.681 \\
&\quad\ \ \textbf{else} \quad \mathsf{NeedlemanWunch[title]} \geqslant 0.733) \bigvee \\
&(\ \mathsf{EditDistance[title]} \geqslant 0.73 \\
&\bigwedge \mathsf{OverlapToken[venue]} \geqslant 0.268)
\end{aligned}$$

This EM rule specifies under which conditions two records are considered as a *match*. Our tool can automatically figure out (1) what similarity function and associated threshold to use for each selected attribute; and more interestingly, (2) under what logic they should be composed, e.g., with conjunctions ($\bigvee$), disjunctions ($\bigwedge$), negations ($\neg$), and **if-then-else** operations.

Below, we focus on the default parameters use for the demo, some evaluation comparison with the prior *state-of-the-art*, and our Web interface for users to tune parameters.

3.1 Default Configuration

Experiment Setup. The system optimizes F-measure on the training data while performing 5-fold cross-validation and compares the results with other interpretable (Decision Trees of depth 3 or 4, SIFI [6]) and non-interpretable (Deep decision trees, SVM) techniques.

Algorithm Parameters. The system uses a generic set of 30 string similarity functions including equality, checking for null values, and 28 others from the Simmetrics package. The internal parameters (cutoffs and heuristics) for synthesis have been empirically selected to balance running time with accuracy of the produced rules.

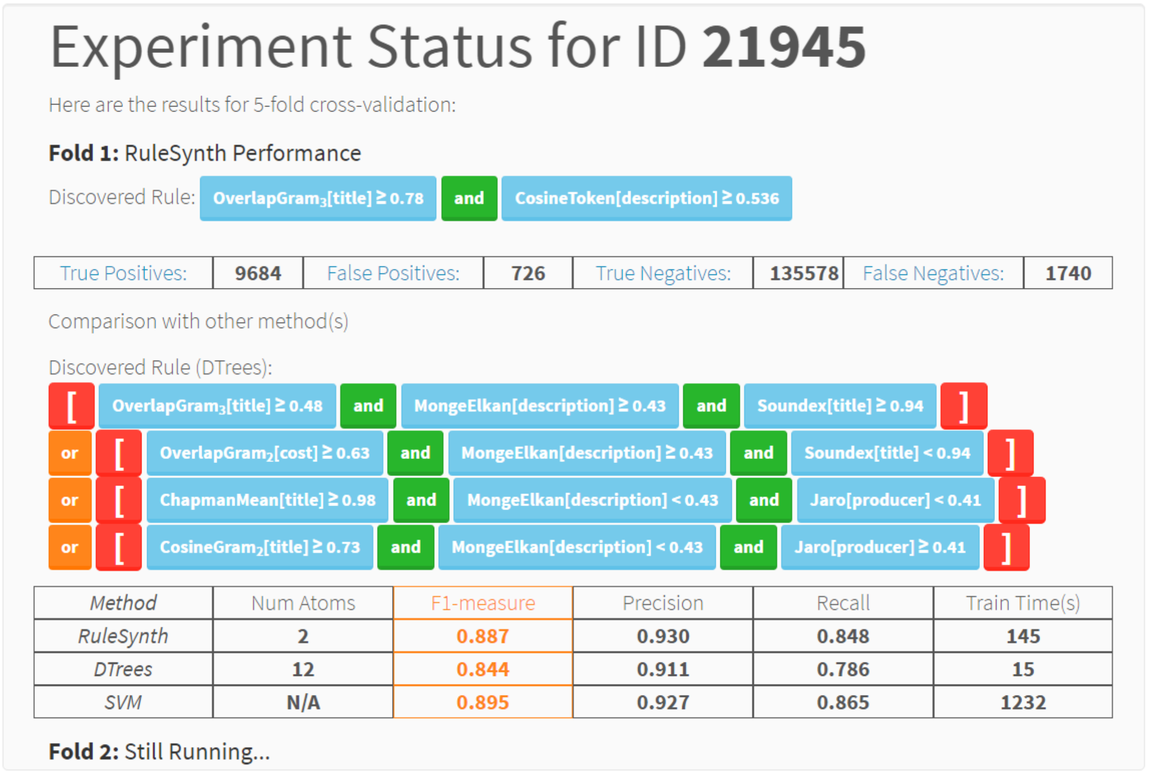
Synthesis Grammar. The system uses a rich interpretable grammar for **GBF**s with conjunction ($\bigwedge$), disjunction ($\bigvee$), **if-then-else**s, and negation ($\neg$) as allowed operations. The grammar is predefined – the user needs neither to provide the grammar, nor to be knowledgeable in program synthesis.

3.2 Evaluation

Interpretability. Besides our results, the attendees can see the performance comparison among the *state-of-the-art* solutions. For every experiment, the results will be displayed in a web-based interface as shown in Figure 2. Generated

	author	title	venue	address	publisher	editor	date	volume	pages
$r \in R$	brodley, c.e., and utgoff, p.e.	multivariate decision trees	machine learning	*null*	*null*	*null*	1995	19(1)	45-77
$s \in S$	carla brodley and paul utgoff.	multivariate trees.	machine learning	*null*	*null*	*null*	1995	19	*null*

Table 2: A Positive Matching Example from Cora Dataset



Experiment Status for ID **21945**

Here are the results for 5-fold cross-validation:

Fold 1: RuleSynth Performance

Discovered Rule: OverlapGram$_3$[title] ≥ 0.78 and CosineToken[description] ≥ 0.536

True Positives:	9684	False Positives:	726	True Negatives:	135578	False Negatives:	1740

Comparison with other method(s)

Discovered Rule (DTrees):

[OverlapGram$_3$[title] ≥ 0.48 and MongeElkan[description] ≥ 0.43 and Soundex[title] ≥ 0.94]
or [OverlapGram$_2$[cost] ≥ 0.63 and MongeElkan[description] ≥ 0.43 and Soundex[title] < 0.94]
or [ChapmanMean[title] ≥ 0.98 and MongeElkan[description] < 0.43 and Jaro[producer] < 0.41]
or [CosineGram$_2$[title] ≥ 0.73 and MongeElkan[description] < 0.43 and Jaro[producer] ≥ 0.41]

Method	Num Atoms	F1-measure	Precision	Recall	Train Time(s)
RuleSynth	2	0.887	0.930	0.848	145
DTrees	12	0.844	0.911	0.786	15
SVM	N/A	0.895	0.927	0.865	1232

Fold 2: Still Running...

Figure 2: Results Interface

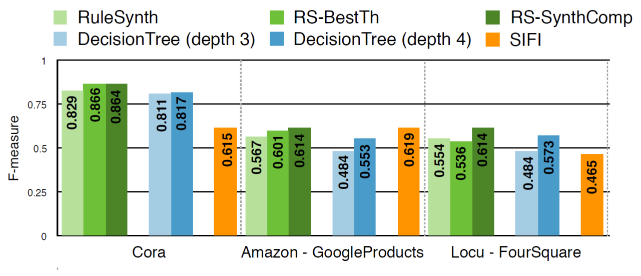


Figure 3: F-measure comparison

rules will be compared against those generated by other techniques to demonstrate the conciseness of our rules. For example, we show the rule generated with a decision tree (φ_{tree}) for the **Cora** dataset (Tables 1 & 2) below:

$$\begin{aligned}\varphi_{tree}: \; & (\ \texttt{OverlapGram}_3[\texttt{title}] \geqslant 0.484 \\ & \quad \bigwedge \texttt{MongeElkan}[\texttt{volume}] \geqslant 0.429 \\ & \quad \bigwedge \texttt{Soundex}[\texttt{title}] \geqslant 0.939) \bigvee \\ & (\ \texttt{OverlapGram}_2[\texttt{pages}] \geqslant 0.626 \\ & \quad \bigwedge \texttt{MongeElkan}[\texttt{volume}] \geqslant 0.429 \\ & \quad \bigwedge \neg\ (\texttt{Soundex}[\texttt{title}] \geqslant 0.939)) \bigvee \\ & (\ \texttt{ChapmanMeanLength}[\texttt{title}] \geqslant 0.978 \\ & \quad \bigwedge \neg\ (\texttt{OverlapGram}_3[\texttt{author}] \geqslant 0.411) \\ & \quad \bigwedge \neg\ (\texttt{MongeElkan}[\texttt{volume}] \geqslant 0.429)) \bigvee \\ & (\ \texttt{CosineGram}_2[\texttt{title}] \geqslant 0.730 \\ & \quad \bigwedge \texttt{OverlapGram}_3[\texttt{author}] \geqslant 0.411 \\ & \quad \bigwedge \neg\ (\texttt{MongeElkan}[\texttt{volume}] \geqslant 0.429))\end{aligned}$$

As shown above, the rule φ_{tree} based on decision tree is much more complicated than the rule φ_{synth} from our technique, as shown earlier in this section.

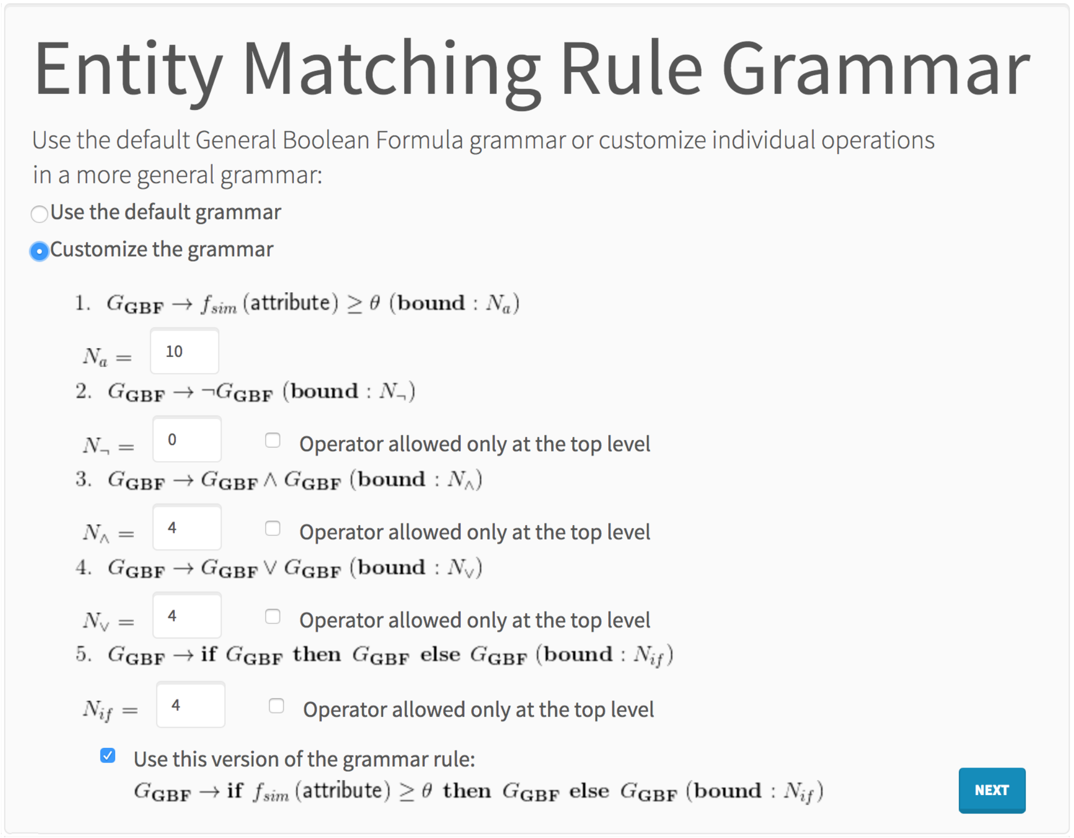
High Performance. Figure 2 also shows the performance of our algorithm and its comparison with other methods when optimizing F-measure. We will show that our tool provides comparable F-measures as other methods. The results obtained for our datasets can be seen in Figure 3. RuleSynth, RS-BestTh, and RS-SynthComp are three variants of our algorithm - RS-SynthComp will be used in the default configuration.

Debugging. Users can download the experiment logs and the records that were misclassified (by clicking on False positives / False negatives in Figure 2) by our algorithm and re-run the experiment based on the insights gained from them with more customizations (as discussed next).

3.3 Customization

The attendees have the opportunities to customize the demo scenarios by modifying the dataset to be used, the grammar feeding the PS engine, and other parameters.

Datasets. The interface in Figure 5 can be used to manipulate existing datasets by (1) introducing errors; (2) reducing the number of positive and/or negative examples; and (3) adding null values in the dataset. These changes make



Entity Matching Rule Grammar

Use the default General Boolean Formula grammar or customize individual operations in a more general grammar:

Use the default grammar [✓ Customize the grammar]

1. $G_{\mathbf{GBF}} \rightarrow f_{sim}(\text{attribute}) \geq \theta\ (\mathbf{bound} : N_a)$

   $N_a =$ 10

2. $G_{\mathbf{GBF}} \rightarrow \neg G_{\mathbf{GBF}}\ (\mathbf{bound} : N_{\neg})$

   $N_{\neg} =$ 0 ☐ Operator allowed only at the top level

3. $G_{\mathbf{GBF}} \rightarrow G_{\mathbf{GBF}} \wedge G_{\mathbf{GBF}}\ (\mathbf{bound} : N_{\wedge})$

   $N_{\wedge} =$ 4 ☐ Operator allowed only at the top level

4. $G_{\mathbf{GBF}} \rightarrow G_{\mathbf{GBF}} \vee G_{\mathbf{GBF}}\ (\mathbf{bound} : N_{\vee})$

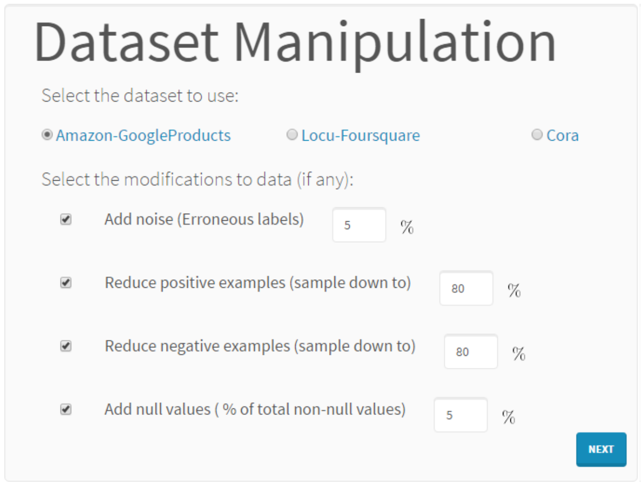
   $N_{\vee} =$ 4 ☐ Operator allowed only at the top level

5. $G_{\mathbf{GBF}} \rightarrow \mathbf{if}\ G_{\mathbf{GBF}}\ \mathbf{then}\ G_{\mathbf{GBF}}\ \mathbf{else}\ G_{\mathbf{GBF}}\ (\mathbf{bound} : N_{if})$

   $N_{if} =$ 4 ☐ Operator allowed only at the top level

   ☑ Use this version of the grammar rule:
   $G_{\mathbf{GBF}} \rightarrow \mathbf{if}\ f_{sim}(\text{attribute}) \geq \theta\ \mathbf{then}\ G_{\mathbf{GBF}}\ \mathbf{else}\ G_{\mathbf{GBF}}\ (\mathbf{bound} : N_{if})$

Figure 4: Modifying EM Rule Grammars



Dataset Manipulation

Select the dataset to use:

[✓ Amazon-GoogleProducts] Locu-Foursquare Cora

Select the modifications to data (if any):

☑ Add noise (Erroneous labels) 5 %

☑ Reduce positive examples (sample down to) 80 %

☑ Reduce negative examples (sample down to) 80 %

☑ Add null values (% of total non-null values) 5 %

Figure 5: Dataset Customization

the EM scenario much harder and enable the attendees to see strengths and limitations of the different approaches.

Optimization Metric. The optimization metric can be set to one of F-measure, Precision, Recall, Accuracy, or can be provided by a user of the tool as native Python code, such as a weighting function that increases the importance of subsets of data to match correctly that are important for the final application, e.g., certain subsets pertaining to specific authors in the **Cora** dataset may need to be matched at a higher accuracy than the others.

Algorithm Parameters. The set of similarity functions can be augmented by adding or replacing custom functions provided as native Python code. The algorithm cutoffs and heuristics can be chosen from a specified range of options.

Synthesis Grammar. A rich set of predefined grammar rules and bounds/constraints on top of them can be modified, e.g., one can limit any operator ($\wedge, \vee, \neg,$ **if**) to occur only at the topmost level, or limit their number of occurrences as they deem fit (Figure 4). This allows experts to see the effects of changing the grammar. For example, richer grammars can express more rules but may result in slower convergence in the synthesis algorithm.

4. REFERENCES

[1] P. A. Bernstein, J. Madhavan, and E. Rahm. Generic schema matching, ten years later. *PVLDB*, 4(11):695–701, 2011.

[2] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *SIGKDD*, pages 39–48, 2003.

[3] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

[4] I. Fellegi and A. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64 (328), 1969.

[5] A. Solar-Lezama. The sketching approach to program synthesis. In *APLAS*, pages 4–13, 2009.

[6] J. Wang, G. Li, J. X. Yu, and J. Feng. Entity matching: How similar is similar. *PVLDB*, 4(10), 2011.